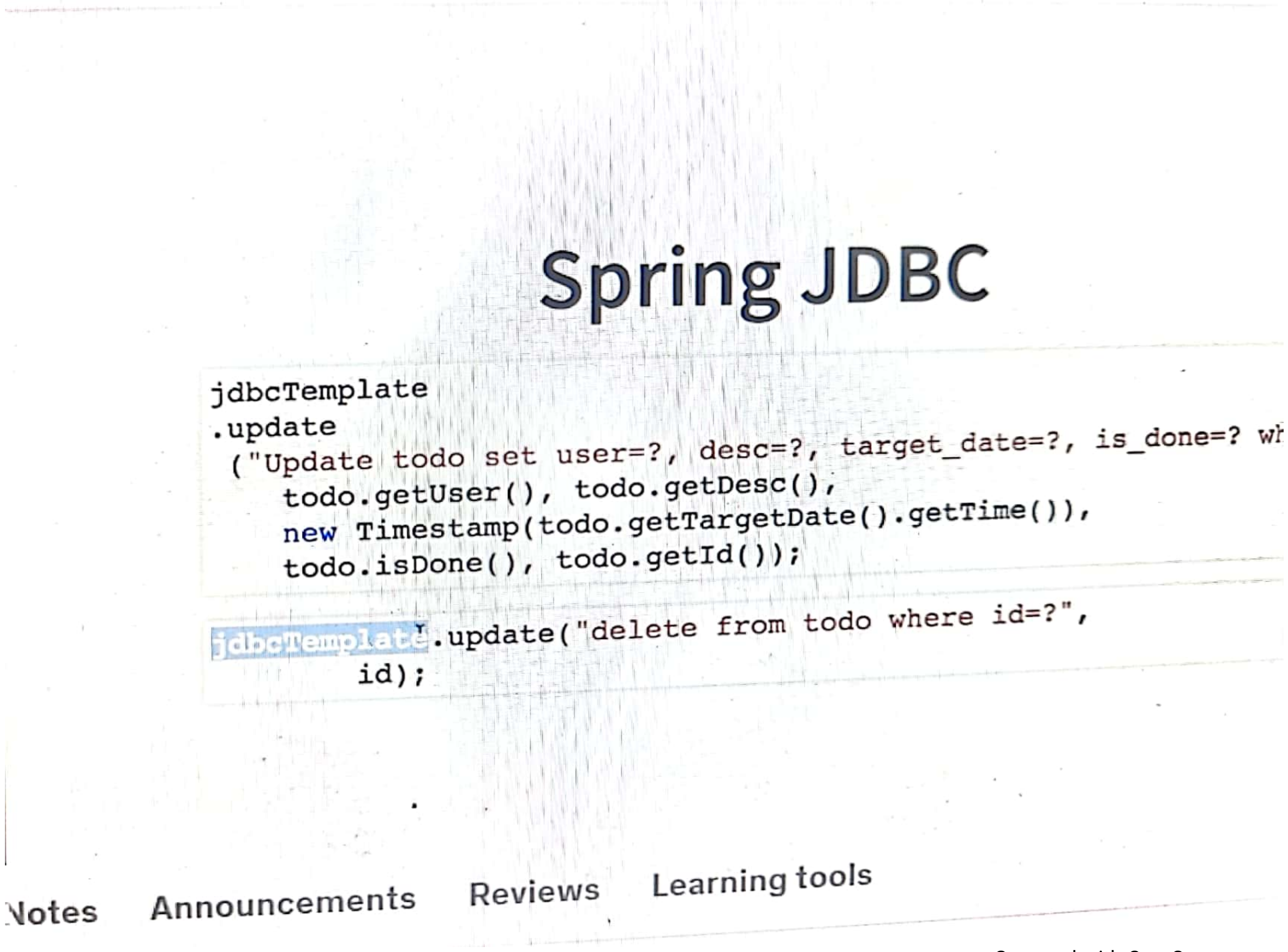
JDBC - Update Todo

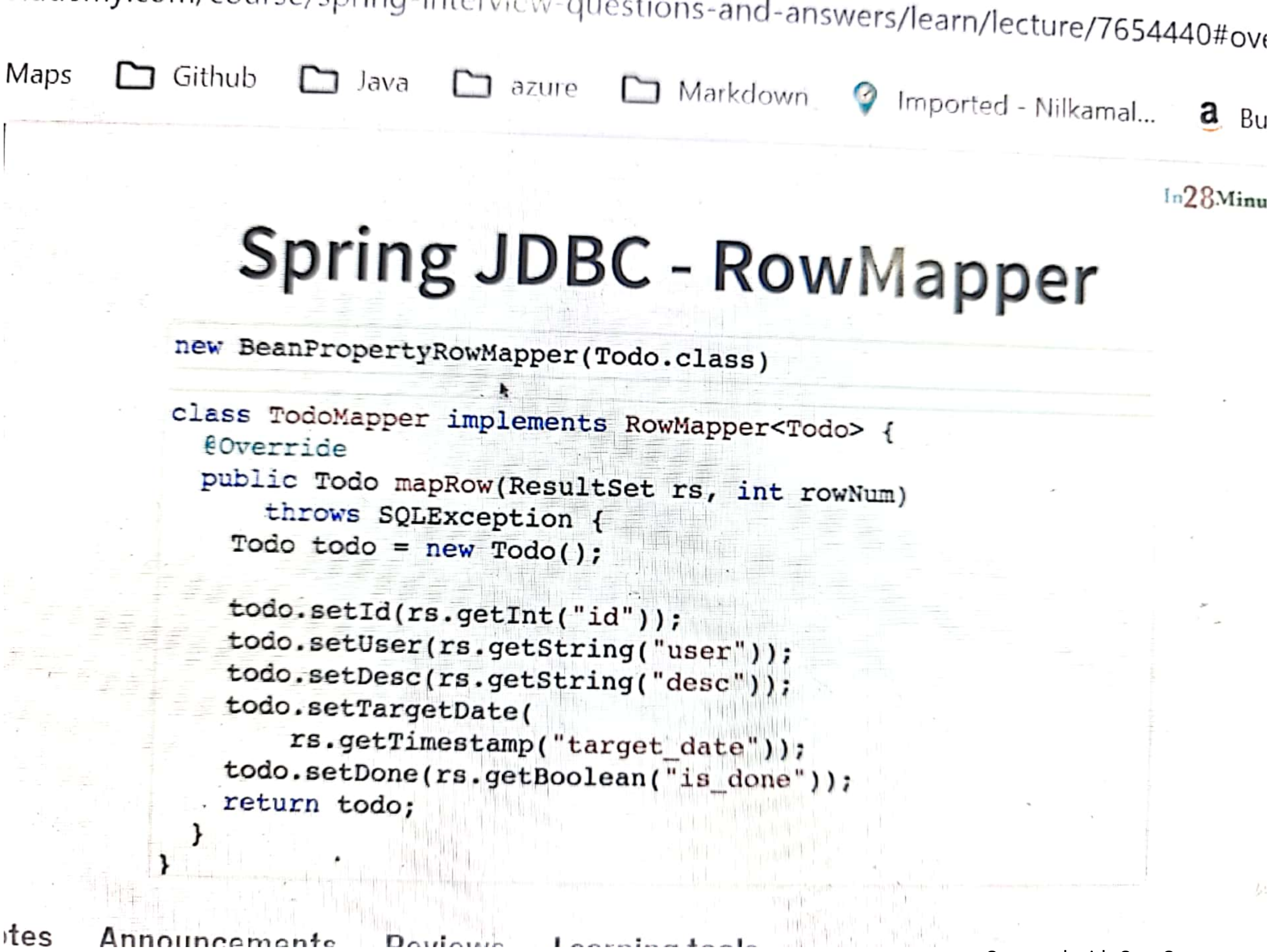
```
Connection connection = datasource.getConnection();

PreparedStatement st = connection.prepareStatement(
  "Update todo set user=?, desc=?, target_date=?, is_done=? wh

st.setString(1, todo.getUser());
st.setString(2, todo.getDesc());
st.setTimestamp(3, new Timestamp(todo.getTargetDate().getTime(
st.setBoolean(4, todo.isDone());
st.setInt(5, todo.getId());

st.execute();
st.close();
connection.close();
```



Spring JDBC

```
jdbcTemplate
.update
 ("Update todo set user=?, desc=?, target_date=?, is_done=? wh
    todo.getUser(), todo.getDesc(),
    new Timestamp(todo.getTargetDate().getTime()),
    todo.isDone(), todo.getId());
```

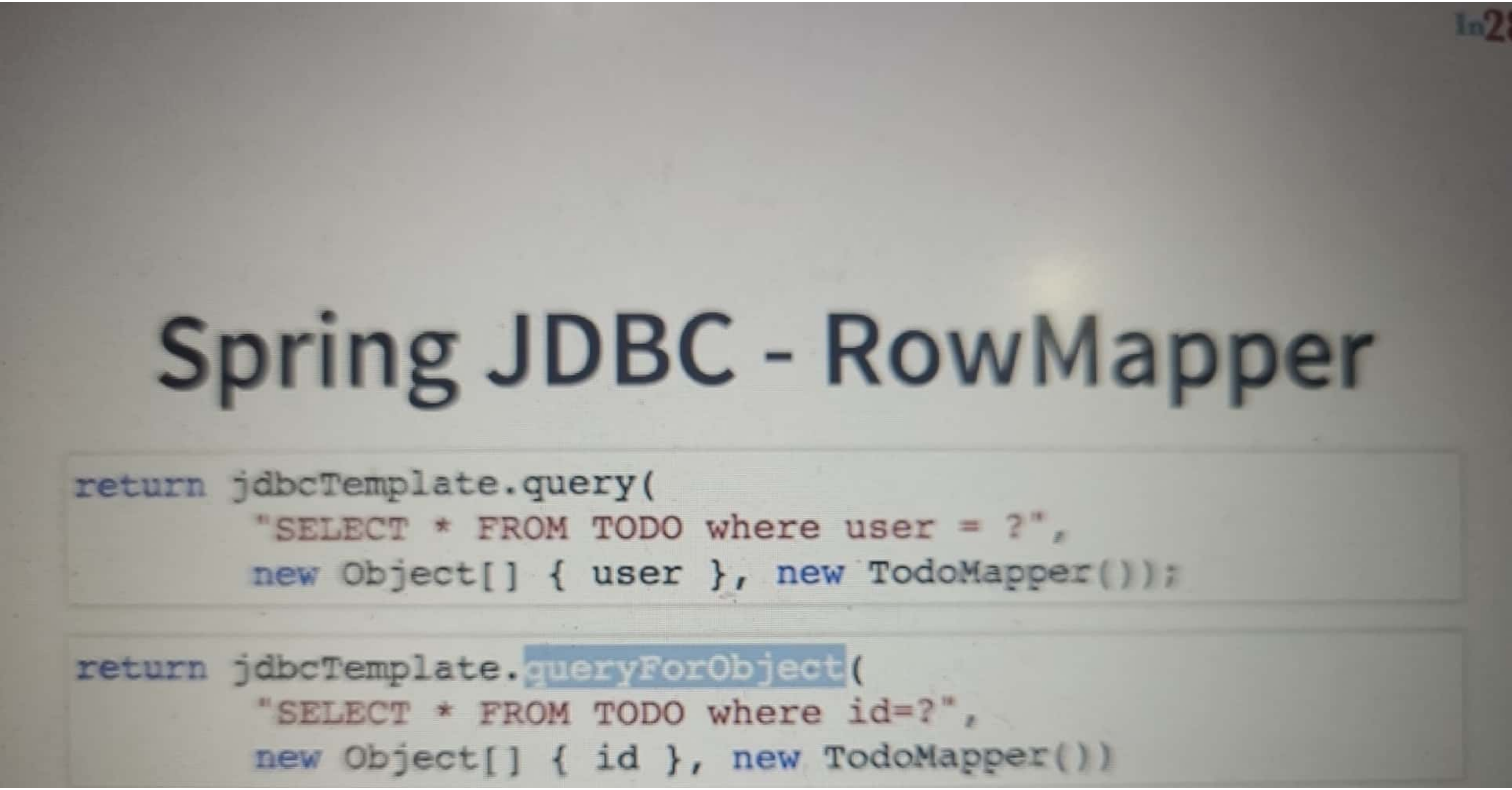
```
jdbcTemplate.update("delete from todo where id=?",
        id);
```



demy.com/course/spring-interview-questions-and-answers/learn/lecture/7654440#ove

Spring JDBC - RowMapper

```
new BeanPropertyRowMapper(Todo.class)

class TodoMapper implements RowMapper<Todo> {
  @Override
  public Todo mapRow(ResultSet rs, int rowNum)
      throws SQLException {
    Todo todo = new Todo();

    todo.setId(rs.getInt("id"));
    todo.setUser(rs.getString("user"));
    todo.setDesc(rs.getString("desc"));
    todo.setTargetDate(
        rs.getTimestamp("target_date"));
    todo.setDone(rs.getBoolean("is_done"));
    return todo;
  }
}
```



Spring JDBC - RowMapper

```
return jdbcTemplate.query(
        "SELECT * FROM TODO where user = ?",
        new Object[] { user }, new TodoMapper());
```

```
return jdbcTemplate.queryForObject(
        "SELECT * FROM TODO where id=?",
        new Object[] { id }, new TodoMapper())
```

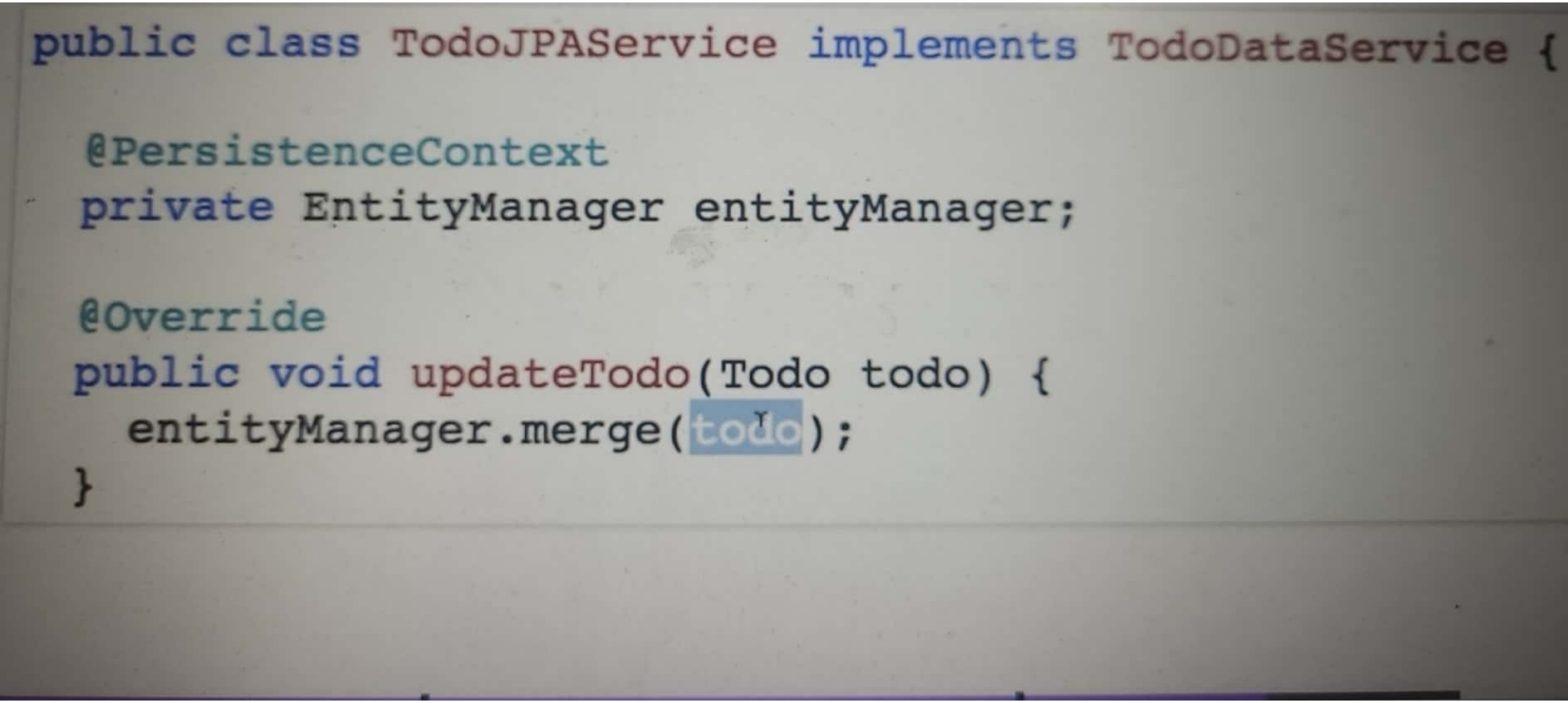
Questions

- What is JPA?
- What is Hibernate?
- How do you define an entity in JPA?
- What is an Entity Manager?
- What is a Persistence Context?

JPA - Update Todo

Defining an entity

```
@Entity
@Table(name = "Todo")
public class Todo {

  @Id
  @GeneratedValue(strategy = GenerationType.IDENTITY)
  private int id;

  private String user;

  private String desc;

  private Date targetDate;

  private boolean isDone;
```



```
public class TodoJPAService implements TodoDataService {

  @PersistenceContext
  private EntityManager entityManager;

  @Override
  public void updateTodo(Todo todo) {
    entityManager.merge(todo);
  }
```

Questions

- How do you map relationships in JPA?
- What are the different types of relationships in JPA?
- How do you define One to One Mapping in JPA?
- How do you define One to Many Mapping in JPA?
- How do you define Many to Many Mapping in JPA?

One to One Relationship

```
@Entity
public class Passport {


   ....
   // Inverse Relationship
   // bi-directional OneToOne relationship
   // Column will not be created in the table
   // Try removing mappedBy = "passport" => You will see that s
   // will be created in passport
   @OneToOne(fetch = FetchType.LAZY, mappedBy = "passport")
   private Student student;
```

```
@Entity
@Table(name = "Student")
public class Student {

   @OneToOne
   private Passport passport;
```

2x

One to Many Relationship

```
@Entity
public class Project {
  @OneToMany(mappedBy = "project")
  private List<Task> tasks;
```

```
@Entity
public class Task {
  @ManyToOne
  @JoinColumn(name="PROJECT_ID")
  private Project project;
```

Many to Many Relationship

```
@Entity
public class Project {

  @ManyToMany
  // @JoinTable(name="STUDENT_PROJ",
  // joinColumns=@JoinColumn(name="STUDENT_ID"),
  // inverseJoinColumns=@JoinColumn(name="PROJECT_ID"))
  private List<Student> students;
```

```
public class Student {
@ManyToMany(mappedBy = "students")
  private List<Project> projects;
```

Questions

- How do you define a datasource in a Spring Context?
- What is the use of persistence.xml
- How do you configure Entity Manager Factory and Transaction Manager?
- How do you define transaction management for Spring – Hibernate integration?

Defining a Data Source

```
#HSQL in-memory db
db.driver=org.hsqldb.jdbcDriver
db.url=jdbc:hsqldb:mem:firstdb
db.username=sa
db.password=
```

```
<bean id="dataSource" class="com.mchange.v2.c3p0.ComboPooledDa
    destroy-method="close">
    <property name="driverClass" value="${db.driver}" />
    <property name="jdbcUrl" value="${db.url}" />
    <property name="user" value="${db.username}" />
    <property name="password" value="${db.password}" />
</bean>
```

Configuring Hibernate

src\main\resources\config\hibernate.properties

```
hibernate.dialect=org.hibernate.dialect.HSQLDialect
hibernate.show_sql=false
hibernate.format_sql=false
hibernate.use_sql_comments=true
```

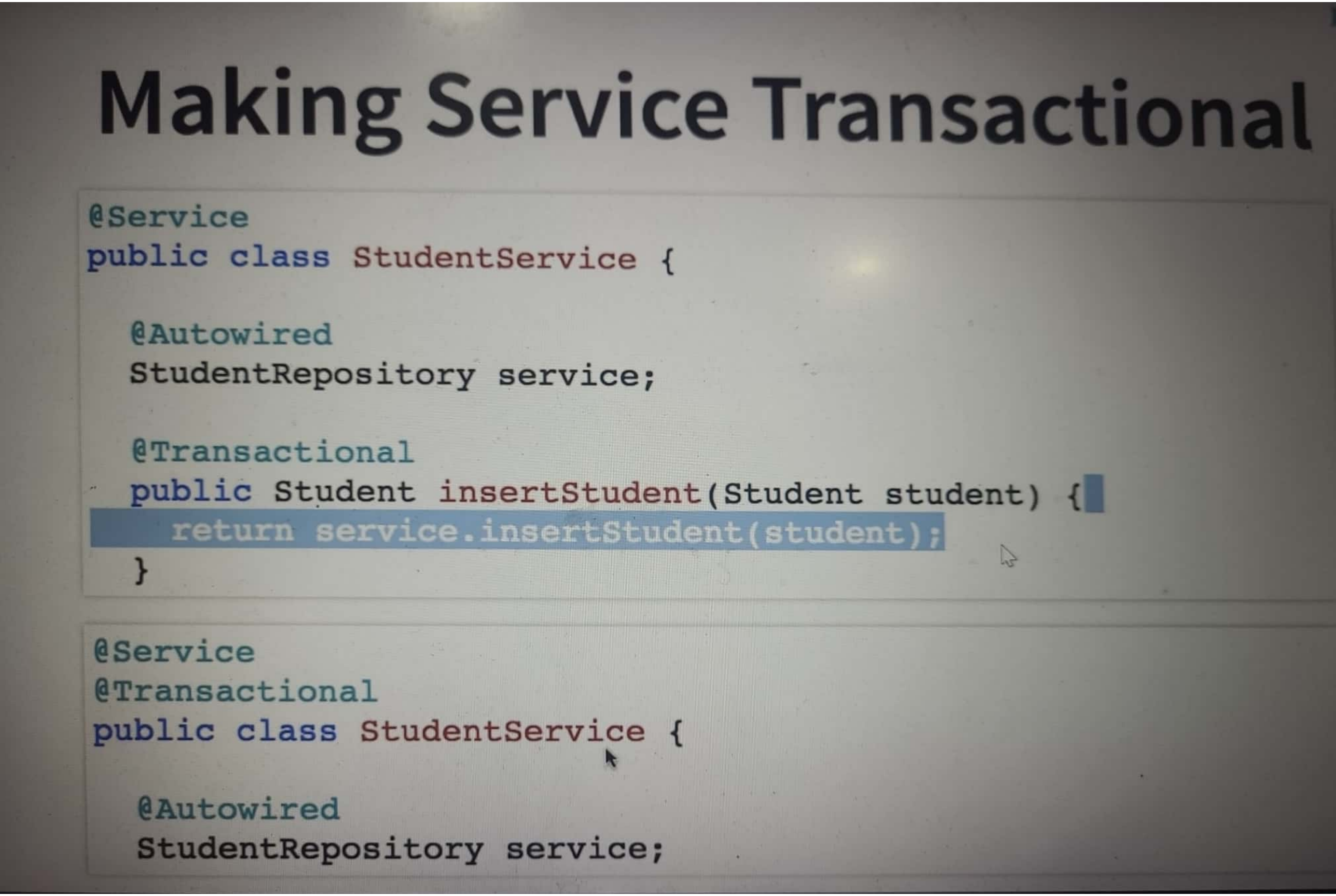
persistence.xml

src\main\resources\META-INF\persistence.xml

```xml
<?xml version="1.0" encoding="UTF-8"?>

<persistence xmlns="http://java.sun.com/xml/ns/persistence"
  version="2.0">
  <persistence-unit name="hsql_pu" transaction-type="RESOURCE_
    <provider>org.hibernate.jpa.HibernatePersistenceProvider</
    <properties>
      <property name="hibernate.dialect" value="org.hibernate.
      <property name="hibernate.connection.url" value="jdbc:hs
      <property name="hibernate.connection.driver_class" value
      <property name="hibernate.connection.username" value="sa
      <property name="hibernate.connection.password" value=""
    </properties>
  </persistence-unit>
</persistence>
```

Configure Entity Manager Factory and Transaction Manager

```
<bean
    class="org.springframework.orm.jpa.LocalContainerEntityMan
    id="entityManagerFactory">
    <property name="persistenceUnitName" value="hsql_pu" />
    <property name="dataSource" ref="dataSource" />
  </bean>

  <bean id="transactionManager" class="org.springframework.orm
    <property name="entityManagerFactory" ref="entityManagerFa
    <property name="dataSource" ref="dataSource" />
  </bean>

  <tx:annotation-driven transaction-manager="transactionManage
```



Making Service Transactional

```
@Service
public class StudentService {

  @Autowired
  StudentRepository service;

  @Transactional
  public Student insertStudent(Student student) {
    return service.insertStudent(student);
  }
```

```
@Service
@Transactional
public class StudentService {

  @Autowired
  StudentRepository service;
```

Questions

- How do you define a datasource in a Spring Context?
- What is the use of persistence.xml
- How do you configure Entity Manager Factory and Transaction Manager?
- How do you define transaction management for Spring – Hibernate integration?

Duplication in JPA Repositories

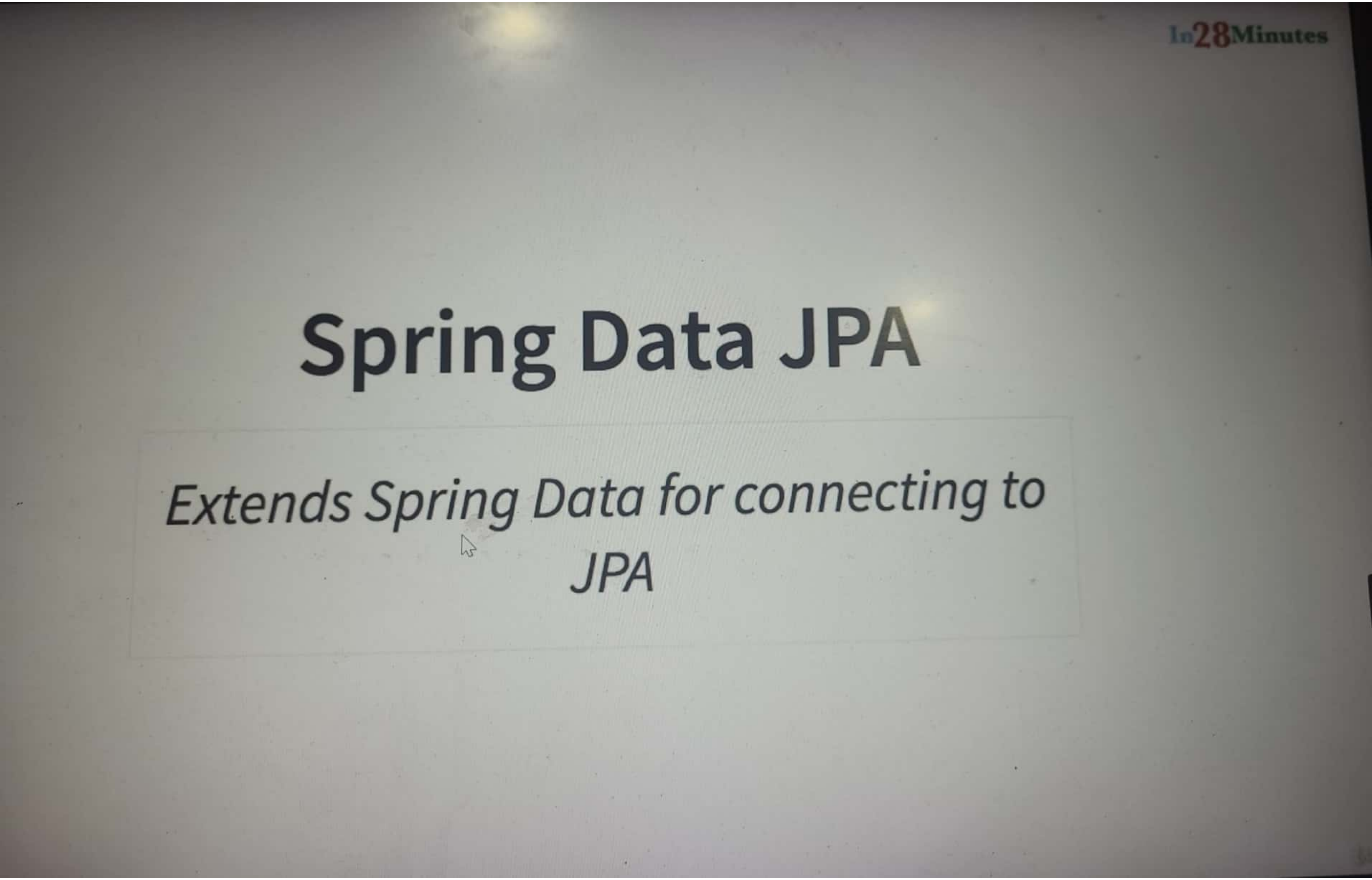
Passport Repository

```
@PersistenceContext
private EntityManager entityManager;

public Passport getPassport(final long id) {
  Passport passport = entityManager
      .find(Passport.class, id);
  return passport;
}

public Passport createPassport(Passport passport) {
  return entityManager.merge(passport);
}
```

Spring Data

Common Abstractions to store and retrieve data from data stores

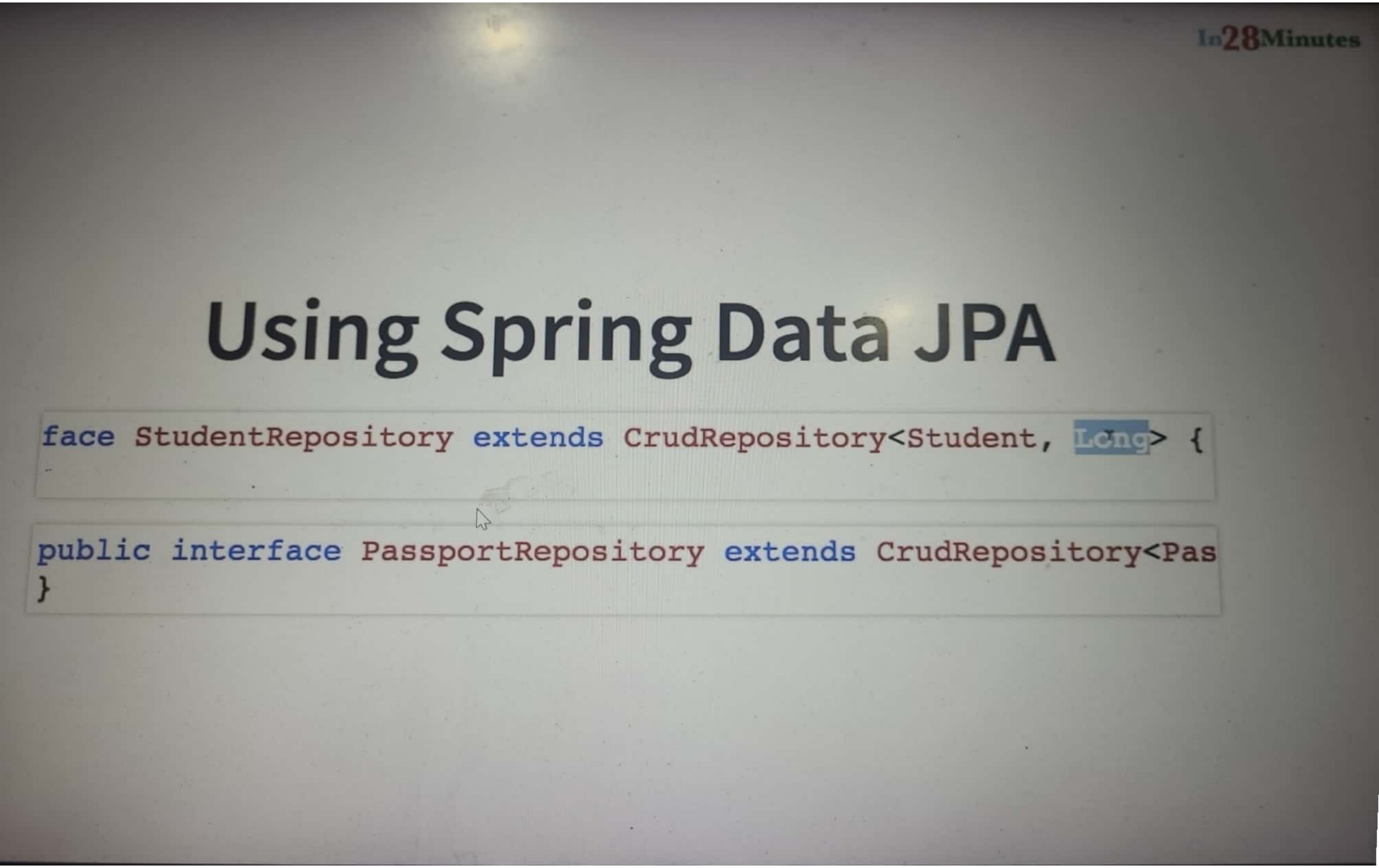
- *Independent of type of data store*



In28Minutes

Spring Data JPA

Extends Spring Data for connecting to JPA



In28Minutes

Using Spring Data JPA

```
face StudentRepository extends CrudRepository<Student, Long> {
```

```
public interface PassportRepository extends CrudRepository<Pas
}
```

In28Minutes

CrudRepository

```
public interface CrudRepository<T, ID> extends Repository<T, I
  <S extends T> S save(S entity);
  Optional<T> findById(ID id);
  boolean existsById(ID id);
  Iterable<T> findAll();
  void deleteById(ID id);
  long count();
  //Other Methods
}
```

Using PagingAndSortingRepository

```
Sort sort = new Sort(Sort.Direction.DESC,"field name");
passportRepository.findAll(sort);
```

```
//Page Size - 10
PageRequest pageable = new PageRequest(0,10);
Page<Passport> page = passportRepository.findAll(pageable);
System.out.println(userPage.getTotalPages());
System.out.println(userPage.nextPageable());
```